28장. Number

1. 표준 빌트인 객체 **Number**는 원시 타입인 숫자를 다룰 때 유용한 프로퍼티와 메서드를 제공

28.1 Number 생성자 함수

1. 표준 빌트인 객체 Number 객체는 생성자 함수 객체다.

   따라서, new 연산자와 함께 호출하여 Number 인스턴스 생성

2. [예제 28-01]

```
const numObj = new Number();
console.log(numObj);  // Number {[[PrimitiveValue]]: 0}
                                                  // {[[PrimitiveValue]
```

- 인수를 전달하지 않고 new 연산자와 함께 호출

3. [예제 28-03]

```
let numObj = new Number('10');
console.log(numObj);  // Number {[[PrimitiveValue]]: 10 }

numObj = new Number('Hello');
console.log(numObj);  // Number {[[PrimitiveValue]]: NaN }
```

- 숫자가 아닌 값 전달 → 숫자로 강제 변환
- 숫자로 변환할 수 없는 값 → NaN

28.2 Number 프로퍼티

28.2.1 Number.EPSILON

1. 부동소수점으로 인해 발생하는 오차를 해결하기 위해 사용

   [예제 28-06]

```
function isEqual(a, b){
    // a - b의 절대값이, Number.EPSILON보다 작으면 같은 수로 인정한다.
    return Math.abs(a - b) < Number.EPSILON;
}

isEqual(0.1 + 0.2, 0.3);  // true
```

28.2.2 Number.MAX_VALUE

1. 자바스크립트에서 표현할 수 있는 가장 큰 양수 값
    - 이보다 더 큰 숫자는 Infinity

28.2.3 Number.MIN_VALUE

1. 자바스크립트에서 표현할 수 있는 가장 작은 양수 값
    - 이보다 더 작은 숫자는 0

28.2.4 Number.MAX_SAFE_INTEGER

1. 자바스크립트에서 안전하게 표현할 수 있는 가장 큰 정수값

28.2.5 Number.MIN_SAFE_INTEGER

1. 자바스크립트에서 안전하게 표현할 수 있는 가장 작은 정수값

28.2.6 Number.POSITIVE_INFINITY

1. 양의 무한대를 나타내는 숫자값 INFINITY와 같음.

28.2.7 Number.NEGATIVE_INFINITY

1. 음의 무한대를 나타내는 숫자값 -INFINITY와 같음.

28.2.8 Number.NaN

1. 숫자가 아님(Not a Number)을 나타내는 숫자 값

2. Number.NaN === window.NaN

28.3 Number 메서드

28.3.1 Number.isFinite

1. 인수로 전달된 숫자값이 정상적인 유한수, 즉 Infinity or -Infinity가 아닌지 검사하여 그 결과를 boolean 값으로 반환

2. 인수가 NaN이면 언제나 false를 반환

3. **빌트인 전역 함수 isFinite**와 **차이**가 있다.
   isFinite는 전달받은 인수를 숫자로 압묵적 타입 변환하여 검사를 수행하지만,
   Number.isFinite는 전달받은 인수를 숫자로 암묵적 타입 변환 X

   [예제 28-16]

```
// Number.isFinite은 인수를 숫자로 암묵적 타입 변환 X
Number.isFinite(null);  // false

// isFinite는 인수를 숫자로 암묵적 타입 변환함. null은 0으로 암묵적 타입
isFinite(null)  // true
```

28.3.2 Number.isInteger

1. 인수로 전달된 숫자값이 정수인지 검사하여 그 결과를 boolean 값으로 반환.
   검사하기 전에 인수를 숫자로 암묵적 타입 변환 X

28.3.3 Number.isNaN

1. 인수로 전달된 숫자값이 NaN인지 검사하여 그 결과를 boolean 값으로 반환.

2. **빌트인 전역 함수 isNaN**와 **차이**가 있다.

   isNaN은 전달받은 인수를 숫자로 압묵적 타입 변환하여 검사를 수행하지만,

   Number.isFinite는 전달받은 인수를 숫자로 암묵적 타입 변환 X

28.3.4 Number.isSafeInteger

1. 인수로 전달된 숫자값이 안전한 정수인지 검사하여 그 결과를 boolean 값으로 반환.

28.3.5 Number.prototype.toExponential

1. 숫자를 지수 표기법으로 변환하여 문자열로 반환
   - 지수 표기법이란, 매우 크거나 작은 숫자를 표기할 때 주로 사용하며 e(Exponent) 앞에 있는 숫자에 10의 n승을 곱하는 형식으로 나타내는 방식

2. 인수로 소수점 이하로 표현할 자릿수를 전달할 수 있다.

```
(77.1234).toExponential();    // "7.71234e+1"
(77.1234).toExponential(4);   // "7.7123e+1"
(77.1234).toExponential(2);   // "7.71e+1"
```

28.3.6 Number.prototype.toFixed

1. 숫자를 반올림하여 문자열로 반환한다.

   [예제 28-26]

```
(12345.6789).toFixed();   // 12346
(12345.6789).toFixed(1);  // 12345.7
(12345.6789).toFixed(2);  // 12345.68
(12345.6789).toFixed(3);  // 12345.679
```

   - 인수를 생략하면 기본값 0 지정

28.3.7 Number.prototype.toPrecision

1. 인수로 전달받은 전체 자릿수까지 유효하도록 나머지 자릿수를 반올림하여 문자열로 반환

28.3.8 Number.prototype.toString

1. 숫자를 문자열로 변환하여 반환. 인수를 생략하면 기본값 10진법이 지정

   [예제 28-28]

```
// 인수 생략 -> 10진수로 반환
(10).toStirng();  // "10"
// 2진수로 반환
(16).toStirng(2);  // "10000"
// 8진수로 반환
(16).toStirng(8);  // "20"
// 16진수로 반환
(16).toStirng(16);  // "10"
```